


```

LL      WW      WW      RRRRRRRR      IIIIII      TTTTTTTTTT      EEEEEEEEEEE
LL      WW      WW      RRRRRRRR      IIIIII      TTTTTTTTTT      EEEEEEEEEEE
LL      WW      WW      RR          RR      II      TT      EE
LL      WW      WW      RR          RR      II      TT      EE
LL      WW      WW      RR          RR      II      TT      EE
LL      WW      WW      RR          RR      II      TT      EE
LL      WW      WW      RRRRRRRR      II      TT      EEEEEEEEE
LL      WW      WW      RRRRRRRR      II      TT      EEEEEEEEE
LL      WW      WW      RR      RR      II      TT      EE
LL      WW      WW      RR      RR      II      TT      EE
LL      WWW      WWW      RR      RR      II      TT      EE
LL      WWW      WWW      RR      RR      II      TT      EE
LLLLLLLLLLLL      WW      WW      RR          RR      IIIIII      TT      EEEEEEEEEEE
LLLLLLLLLLLL      WW      WW      RR          RR      IIIIII      TT      EEEEEEEEEEE

```

```

LL                      IIIII
LL                      IIIII
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LLLLLLLLLLLL           IIIII
LLLLLLLLLLLL           IIIII

SSSSSSSS
SSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSS
SSSSSSSS

```

```
0001 0 %TITLE 'EDTSLWRITE - write to a file'
0002 0 MODULE EDTSLWRITE (
0003 0 IDENT = 'V04-000'
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *****
0028 1
0029 1
0030 1
0031 1 ++
0032 1 FACILITY: EDT -- The DEC Standard Editor
0033 1
0034 1 ABSTRACT:
0035 1
0036 1 This module executes the line mode EXIT and WRITE commands.
0037 1
0038 1 ENVIRONMENT: Runs at any access mode - AST reentrant
0039 1
0040 1 AUTHOR: Bob Kushlis, CREATION DATE: February 3, 1978
0041 1
0042 1 MODIFIED BY:
0043 1
0044 1 1-001 - Original. DJS 30-JAN-1981. This module was created by
0045 1 extracting the routines WRITE_FILE, EDT$SEXI_CMD, and EDT$SWR_CMD
0046 1 from the module EXEC.BLI.
0047 1 1-002 - Regularize headers. JBS 23-Mar-1981
0048 1 1-003 - Change WRITE_COM to EDT$SWR_CMD. JBS 30-Mar-1981
0049 1 1-004 - Use the new message codes. JBS 04-Aug-1981
0050 1 1-005 - Don't write out summary when flag clear. STS 05-Oct-1981
0051 1 1-006 - Convert open output file to use EDT$FILEIO. STS 13-Jan-1982
0052 1 1-007 - Remove division from line number calculations. SMB 15-Jan-1982
0053 1 1-008 - Convert reads and writes to use EDT$FILEIO. STS 15-Jan-1982
0054 1 1-009 - Add capability for 15 digit line numbers. SMB 19-Jan-1982
0055 1 1-010 - pass sequence numbers by descriptors. STS 20-Jan-1981
0056 1 1-011 - Change the checking of sequence/nosequence. STS 21-Jan-1982
0057 1 1-012 - Fix size bugs related to 15 digit sequences & increments. SMB 26-Jan-1982
```


58	0058	1	1-013	- Remove original line numbers. SMB 28-Jan-1982
59	0059	1	1-014	- Fix bug in writing with /SEQ. SMB 30-Jan-1982
60	0060	1	1-015	- Add range checking on seq.# and incr. SMB 5-Feb-1982
61	0061	1	1-016	- Take out call to edt\$\$get_fnam. STS 10-Feb-1982
62	0062	1	1-017	- Change line number division to a routine call. SMB 11-Feb-1982
63	0063	1	1-018	- Format filename for output. STS 12-Feb-1982
64	0064	1	1-019	- Pass filename to edt\$\$fiopn err. STS 26-Feb-1982
65	0065	1	1-020	- Add literals for callable EDT. STS 08-Mar-1982
66	0066	1	1-021	- Use the input file name as the output name, if permitted and there is no output file name. JBS 25-Mar-1982
67	0067	1		
68	0068	1	1-022	- The file I/O routines return a status. JBS 26-Mar-1982
69	0069	1	1-023	- Correct a typo in edit 1-022. JBS 27-Mar-1982
70	0070	1	1-024	- Minor edits, plus remove code that doesn't seem to be executed. SMB 30-Mar-1982
71	0071	1	1-025	- Rearrange where file name is stored so we get "real" name instead of "temp" filename. SMB 31-Mar-1982
72	0072	1		
73	0073	1	1-026	- Add code for CONTROL C check during record writing. SMB 08-Apr-1982
74	0074	1	1-027	- Print messages on CLOSE errors. JBS 12-Apr-1982
75	0075	1	1-028	- Convert PDP-11 command file names to uppercase. SMB 13-Apr-1982
76	0076	1	1-029	- Clear PREV_RANGE when creating a range block for EXIT. JBS 15-Apr-1982
77	0077	1	1-030	- Don't set up any message for nosummary. STS 16-Apr-1982
78	0078	1	1-031	- Change NO FILE error message and processing for WRITE. SMB 21-Apr-1982
79	0079	1	1-032	- Save original buffer address. STS 10-May-1982
80	0080	1	1-033	- Make minor modifications based on code review input. SMB 24-May-1982
81	0081	1	1-034	- Set a flag if control C actually aborts something. JBS 24-May-1982
82	0082	1	1-035	- Fix bug with explicit file names. SMB 25-May-1982
83	0083	1	1-036	- Pass default file name using RHB parameter. JBS 15-Jun-1982
84	0084	1	1-037	- Stop working message before printing messages to screen. SMB 22-Jun-1982
85	0085	1	1-038	- Stop processing if bad select range. SMB 01-Jul-1982
86	0086	1	1-039	- Use EDT\$\$FMT_CRLF instead of EDT\$\$OUT_FMTBUF. JBS 05-Jul-1982
87	0087	1	1-040	- Change write file messages. SMB 13-Jul-1982
88	0088	1	1-041	- Give message for write errors. STS 21-Jul-1982
89	0089	1	1-042	- Don't put out error messages on 11's STS 22-Jul-1982
90	0090	1	1-043	- Restore screen width before displaying file name on EXIT. SMB 29-Jul-1982
91	0091	1	1-044	- Check the abort write flag after reads. STS 05-Aug-1982
92	0092	1	1-045	- Make use of the bit masks for the options word. STS 17-Aug-1982
93	0093	1	1-046	- Fix up the references to the EDT\$M_ symbols. JBS 23-Aug-1982
94	0094	1	1-047	- Put EDT\$\$TST_EOB in line. STS 22-Sep-1982
95	0095	1	1-048	- Change to use new addline/subline macro. STS 1-Oct-1982
96	0096	1	1-049	- Put code for edt\$\$rng_posfrst in line. STS 11-Oct-1982
97	0097	1	1-050	- Modify to use new compare macro. STS 10-Oct-1982
98	0098	1	1-051	- Avoid str\$copy where possible to increase efficiency. STS 10-Nov-1982
99	0099	1	1-052	- Add conditional for WPS support. JBS 10-Feb-1983
100	0100	1	1-053	- Only set screen width if it has changed from original. SMB 17-Feb-1983
101	0101	1	1-054	- Give a message if requested to do so. JBS 24-Mar-1983
102	0102	1	1-055	- Set a flag if control C actually aborts something. JBS 08-Apr-1983
103	0103	1	1-056	- Delete the output file if we fail to write a record. JBS 02-May-1983
104	0104	1	1-057	- Rearrange the delete output file logic to make this module smaller on PDP-11s, and to always give the "no output file created" message when we delete the output file. JBS 02-May-1983
105	0105	1		
106	0106	1		
107	0107	1	1-058	- Because of search lists on VMS, don't use the input file name. JBS 29-Jul-1983
108	0108	1	--	
109	0109	1		

EDT\$WRITE
V04-000

EDT\$WRITE - write to a file
Declarations

D 1
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1
Page 3
(2)

```
111 0110 1 %SBTTL 'Declarations'
112 0111 1
113 0112 1 TABLE OF CONTENTS:
114 0113 1
115 0114 1
116 0115 1 REQUIRE 'EDT$SRC:TRAROUNAM';
117 0554 1
118 0555 1 FORWARD ROUTINE
119 0556 1 WRITE_FILE,
120 0557 1 EDT$EXI_CMD : NOVALUE,
121 0558 1 EDT$WR_CMD : NOVALUE;
122 0559 1
123 0560 1
124 0561 1 INCLUDE FILES:
125 0562 1
126 0563 1
127 0564 1 REQUIRE 'EDT$SRC:EDTREQ';
128 0699 1
129 L 0700 1 %IF %BLISS (BLISS32)
130 0701 1 %THEN
131 0702 1
132 0703 1 REQUIRE 'EDT$SRC:SYSSYM';
133 0733 1
134 0734 1 %FI
135 0735 1
136 0736 1 LIBRARY 'EDT$SRC:SUPPORTS';
137 0737 1
138 0738 1
139 0739 1 MACROS:
140 0740 1
141 0741 1 NONE
142 0742 1
143 0743 1 EQUATED SYMBOLS:
144 0744 1
145 0745 1
146 0746 1 EXTERNAL LITERAL
147 0747 1 EDT$M_NOOUTPUT,
148 0748 1 EDT$K_WRITE_FILE,
149 0749 1 EDT$K_OUTPUT_FILE,
150 0750 1 EDT$K_OPEN_OUTPUT_SEQ,
151 0751 1 EDT$K_OPEN_OUTPUT_NOSEQ,
152 0752 1 EDT$K_PUT,
153 0753 1 EDT$K_CLOSE,
154 0754 1 EDT$K_CLOSE_DEL;
155 0755 1
156 0756 1
157 0757 1 OWN STORAGE:
158 0758 1
159 0759 1 NONE
160 0760 1
161 0761 1 EXTERNAL REFERENCES:
162 0762 1
163 0763 1 In the routines
```

```
! Write an output file
! Process the EXIT command
! Process the WRITE command
```

ED
VO

EDT\$LWRITE
V04-000

EDT\$LWRITE - write to a file
WRITE_FILE - write on a file

E 1
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1
Page 4
(3)

ED
V0

```

165 0764 1 %SBTTL 'WRITE_FILE - write on a file'
166 0765 1 ROUTINE WRITE_FILE (
167 0766 1     RANGE,
168 0767 1     CHECK
169 0768 1 ) =
170 0769 1
171 0770 1 ++
172 0771 1 FUNCTIONAL DESCRIPTION:
173 0772 1
174 0773 1     This routine is used by the WRITE and EXIT commands to write
175 0774 1     an output file. The /SEQ switch determines whether sequence
176 0775 1     numbers should be written.
177 0776 1
178 0777 1 FORMAL PARAMETERS:
179 0778 1
180 0779 1     RANGE                the range block for the range to be written.
181 0780 1
182 0781 1     CHECK                a flag indicating that the consistency check should be done
183 0782 1                     (set if we are exiting, clear otherwise). Also permits defaulting
184 0783 1                     of the file name to the output or input file name.
185 0784 1
186 0785 1 IMPLICIT INPUTS:
187 0786 1
188 0787 1     EDT$SG_OUT_NAMLEN
189 0788 1     EDT$SA_OUT_NAM
190 0789 1     EDT$SG_INP_NAMLEN
191 0790 1     EDT$SA_INP_NAM
192 0791 1     EDT$SV_OPTIONS
193 0792 1     EDT$SA_CUR_BUF
194 0793 1     EDT$SL_IO_VFCHD
195 0794 1     EDT$SL_LNO_ZERO
196 0795 1     EDT$SL_LNOD
197 0796 1     EDT$SA_FMT_CUR
198 0797 1     EDT$ST_FMT_BUF
199 0798 1     EDT$SG_FMT_LNPOS
200 0799 1     EDT$SL_LNO_VFCMAX
201 0800 1     EDT$SA_WK_CN
202 0801 1     EDT$SA_SUMRY
203 0802 1     EDT$SA_EXE_CURCMD
204 0803 1     EDT$SG_EXE_SBITS
205 0804 1     EDT$SZ_EXE_SBLK
206 0805 1     EDT$SG_WRITE_MSG
207 0806 1     EDT$SG_ABT_WRITE
208 0807 1
209 0808 1 IMPLICIT OUTPUTS:
210 0809 1
211 0810 1     EDT$SG_EXE_SBITS
212 0811 1     EDT$SG_CC_DONE
213 0812 1     EDT$SG_WRITE_MSG
214 0813 1     EDT$SG_ABT_WRITE
215 0814 1
216 0815 1 ROUTINE VALUE:
217 0816 1
218 0817 1     0 = no write took place
219 0818 1     1 = file written successfully
220 0819 1
221 0820 1 SIDE EFFECTS:
```

```

222 0821 1 |
223 0822 1 | Types an error message if the file is not properly written.
224 0823 1 | If the consistency check fails, arranges to save the journal file.
225 0824 1 |
226 0825 1 |
227 0826 1 |
228 0827 2 BEGIN
229 0828 2
230 0829 2 EXTERNAL ROUTINE
231 0830 2 EDTSSC_SETWID, ! Reset terminal width
232 0831 2 EDTSSC_POSCIF, ! Absolute cursor positioning
233 0832 2 EDTSSSTOP_WKINGMSG, ! Stop working message
234 0833 2 EDTSSCHK_CC, ! check for CONTROL/C typed
235 0834 2 EDTSSLDIV,
236 0835 2 EDTSSCALLFIO, ! sets up handler and calls general file routine
237 0836 2 EDTSSFMT_CH, ! Put a character in the format buffer
238 0837 2 EDTSSFMT_STRCNT : NOVALUE,
239 0838 2 EDTSSFMT_CRLF, ! Terminate an output line
240 0839 2 EDTSSFMT_MSG, ! Put a message in the format buffer
241 0840 2 EDTSSNXT_LNRNG,
242 0841 2 EDTSSRNG_REPOS,
243 0842 2 EDTSSFIOPN_ERR, ! Print a file I/O error
244 0843 2
245 L 0844 2 %IF %BLISS (BLISS32)
246 0845 2 %THEN
247 0846 2
248 0847 2 EXTERNAL ROUTINE
249 0848 2 EDTSSFMT_STR,
250 0849 2 STR$FREET_DX; ! deallocates dynamic descriptors
251 0850 2
252 U 0851 2 %ELSE
253 0852 2
254 0853 2 EXTERNAL ROUTINE
255 0854 2 EDTSSGET_FNAM;
256 0855 2
257 0856 2 EXTERNAL
258 0857 2 EDTSSG_INP_NAMLEN, ! Length of input file name
259 0858 2 EDTSSA_INP_NAM; ! Address of input file name
260 0859 2
261 U 0860 2 %FI
262 0861 2
263 0862 2 EXTERNAL
264 0863 2 EDTSSG_TI_WID, ! Terminal's width
265 0864 2 EDTSSG_ABT_WRITE, ! 1 = abort output
266 0865 2 EDTSSG_SAV_TIWID,
267 0866 2 EDTSSG_MESSAGE_LINE,
268 0867 2 EDTSSA_IO_FNAM,
269 0868 2 EDTSSG_EXT_MOD,
270 0869 2 EDTSSG_OUT_NAMLEN, ! Length of output file name
271 0870 2 EDTSSA_OUT_NAM, ! Address of output file name
272 0871 2 EDTSSG_RNG_FRSTLN,
273 0872 2 EDTSS2_RNG_ORIGPOS : POS_BLOCK,
274 0873 2 EDTSSV_OPTIONS : BITVECTOR [32], ! bit 3 = /NOOUTPUT
275 0874 2 EDTSSA_CUR_BUF : REF TBCB_BLOCK,
276 0875 2 EDTSSL_IO_VFCHD : WORD,
277 0876 2 EDTSSL_LNO_ZERO,
278 0877 2 EDTSSL_LNO_VFCMAX : LN_BLOCK, ! Value 6.5535 * (10**9)
```



```
279      0878      2      EDTSSL_LNOO : LNOVECTOR [14],      ! powers of ten
280      0879      2      EDTSSA_FMT_CUR,      ! current position in format buffer
281      0880      2      EDTSSG_FMT_LNPOS,
282      0881      2      EDTSS_T_FMT_BUF,      ! address of format buffer
283      0882      2
284      L 0883      2      %IF SUPPORT_WPS
285      0884      2      %THEN
286      0885      2      EDTSSG_SUMRY,      ! flag indicating whether to type out summary
287      0886      2      %FI
288      0887      2
289      0888      2      EDTSSZ_EOB_LN,
290      0889      2      EDTSSA_WK_CN : REF LIN_BLOCK,
291      0890      2      EDTSSA_EXE_CURCMD : REF NODE_BLOCK,      ! Pointer to the current command.
292      0891      2      EDTSSG_EXE_SBITS,      ! The options switches.
293      0892      2      EDTSSZ_EXE_SBLK : REF NODE_BLOCK,      ! The option switch value block.
294      0893      2      EDTSSG_CC_DONE,      ! Set to 1 if control C actually aborts something
295      0894      2      EDTSSG_WRITE_MSG;      ! Message to print; 1 = no message
296      0895      2
297      P 0896      2      MESSAGES ((NOFILSPC, NOFILWRT, CONCHKFLD, ERRROUTFIL, WRIFILCRE, OUTFILCRE, SEQNUMOV, SEQINCROV, OUTFILCL
298      0897      2      WRIFILCLO));
299      0898      2
300      0899      2      LOCAL
301      0900      2      CLOSEMSG,      ! File close message
302      0901      2      OPNMSG,      ! File open message
303      0902      2      FILECODE,      ! code for what kind of open
304      0903      2      FILESTRM,      ! code for which kind of file
305      0904      2      FILE_DESC : BLOCK [8, BYTE],      ! descriptor for file name
306      0905      2      RHB_DESC : BLOCK [8, BYTE],      ! descriptor for header info
307      0906      2      STAT_DESC : BLOCK [8, BYTE],      ! descriptor for puts
308      0907      2      STATUS,      ! Status of open or close attempt
309      0908      2      L_COUNT : LN_BLOCK,      ! Number of lines written.
310      0909      2      C_COUNT,      ! Number of characters written
311      0910      2      NAME,      ! File name string pointer
312      0911      2      NAME_LEN,      ! File name string length
313      0912      2      DIGIT,      ! Holds integer result from division
314      0913      2      LINNO : LN_BLOCK,      ! 48-bit line number
315      0914      2      SEQ,      ! Flag indicating sequenced
316      0915      2      SEQ_NUM,      ! Current sequence number
317      0916      2      SEQ_INCR;      ! Sequence number increment
318      0917      2
319      L 0918      2      %IF %BLISS (BLISS32)
320      0919      2      %THEN
321      0920      2      RHB_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
322      0921      2      RHB_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
323      0922      2      STAT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
324      0923      2      STAT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
325      0924      2      FILE_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
326      0925      2      FILE_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
327      0926      2      %FI
328      0927      2
329      0928      2      RHB_DESC [DSC$A_POINTER] = 0;
330      0929      2      RHB_DESC [DSC$W_LENGTH] = 0;
331      0930      2      FILE_DESC [DSC$A_POINTER] = 0;
332      0931      2      FILE_DESC [DSC$W_LENGTH] = 0;
333      0932      2      !
334      0933      2      NAME_LEN = 0;
335      0934      2      !+
```


EDTSLWRITE
V04-000

EDTSLWRITE - write to a file
WRITE_FILE - write on a file

H 1
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1 Page 7
(3)

```

336 0935 2 | Position to the front of the range.
337 0936 2 | -
338 0937 2 | EDT$$G_RNG_FRSTLN = 1;
339 0938 2 | EDT$$CPY_MEM (POS_SIZE, .EDT$$A_CUR_BUF, EDT$$Z_RNG_ORIGPOS);
340 0939 2 |
341 0940 2 | IF ( NOT EDT$$RNG_REPOS (.RANGE)) THEN RETURN (0);
342 0941 2 |
343 0942 2 | +
344 0943 2 | Check for an explicit file specification.
345 0944 2 | -
346 0945 2 |
347 0946 2 | IF (.EDT$$A_EXE_CURCMD [FSPCLEN] NEQ 0)
348 0947 2 | THEN
349 0948 2 | BEGIN
350 0949 2 | NAME = .EDT$$A_EXE_CURCMD [FILSPEC];
351 0950 2 | NAME_LEN = .EDT$$A_EXE_CURCMD [FSPCLEN];
352 0951 2 | END;
353 0952 2 |
354 0953 2 | +
355 0954 2 | If no file name is specified and this is EXIT, use the output file name.
356 0955 2 | -
357 0956 2 |
358 0957 2 | IF ((.CHECK NEQ 0) AND (.NAME_LEN EQL 0))
359 0958 2 | THEN
360 0959 2 | BEGIN
361 0960 2 | NAME = .EDT$$A_OUT_NAM;
362 0961 2 | NAME_LEN = .EDT$$G_OUT_NAMLEN;
363 0962 2 | END;
364 0963 2 |
365 0964 2 | +
366 0965 2 | On VMS, because of search lists, it is possible for an input open to resolve
367 0966 2 | to a different directory than an output open with the same file name. Therefore,
368 0967 2 | if we want the output file to be returned to the same directory as the input file
369 0968 2 | we must depend on using the resultant file name from the input open to specify
370 0969 2 | where to put the output file, rather than being able to just re-use the input
371 0970 2 | file name.
372 0971 2 | At this point we reject the file specification only if it is empty and either
373 0972 2 | this is a WRITE command or /NOOUTPUT has been specified (or implied by /READ_ONLY).
374 0973 2 | -
375 0974 2 |
376 0975 2 | IF ((.NAME_LEN EQL 0) AND ((.CHECK EQL 0) OR ((.EDT$$V_OPTIONS AND EDT$M_NOOUTPUT) NEQ 0)))
377 0976 2 | THEN
378 0977 2 | BEGIN
379 0978 2 | EDT$$FMT_MSG (EDT$ _NOFILSPC);
380 0979 2 | RETURN (0);
381 0980 2 | END;
382 0981 2 |
383 0982 2 | +
384 0983 2 | PDP-11 systems do not have search lists, so we can use the input file name if the
385 0984 2 | output name is empty. Doing this simplifys the process of constructing the printable
386 0985 2 | form of the name for the summary message.
387 0986 2 | -
388 0987 2 |
389 L 0988 2 | XIF ( NOT XBLISS (BLISS32))
390 U 0989 2 | XTHEN
391 U 0990 2 |
392 U 0991 2 | IF (.NAME_LEN EQL 0)
```

```
393 U 0992 2 THEN
394 U 0993 2 BEGIN
395 U 0994 2 NAME = .EDTSSA_INP_NAM;
396 U 0995 2 NAME_LEN = .EDTSSG_INP_NAMLEN;
397 U 0996 2 END;
398 U 0997 2
399 U 0998 2 XF1
400 U 0999 2
401 U 1000 2 +
402 U 1001 2 - Check for the /SEQ option.
403 U 1002 2
404 U 1003 2
405 U 1004 2 IF (SEQ = .EDTSSG_EXE_SBITS<OPB_SEQ>)
406 U 1005 2 THEN
407 U 1006 2 BEGIN
408 U 1007 2 +
409 U 1008 2 - Determine the actual sequence start and increment by reducing them by 10**5
410 U 1009 2
411 U 1010 2 SEQ_NUM = 0;
412 U 1011 2 SEQ_INCR = 0;
413 U 1012 2 MOVELINE (EDTSSZ_EXE_SBLK [SW_VAL1], LINNO); ! Starting Value
414 U 1013 2 +
415 U 1014 2 - Error checks for sequence number and increment (must be less than 65536)
416 U 1015 2
417 U 1016 2
418 U 1017 2 IF (CMPLNO (LINNO, EDTSSL_LNO_VFCMAX) GTR 0)
419 U 1018 2 THEN
420 U 1019 2 BEGIN
421 U 1020 2 EDTSSFMT MSG (EDT$_SEQNUMOV);
422 U 1021 2 RETURN (0)
423 U 1022 2 END;
424 U 1023 2
425 U 1024 2 DECR I FROM 9 TO 5 DO
426 U 1025 2 BEGIN
427 U 1026 2 SEQ_NUM = .SEQ_NUM*10;
428 U 1027 2 EDTSLDIV (LINNO, DIGIT, .1);
429 U 1028 2 SEQ_NUM = .SEQ_NUM + .DIGIT;
430 U 1029 2 END;
431 U 1030 2
432 U 1031 2 MOVELINE (EDTSSZ_EXE_SBLK [SW_VAL2], LINNO); ! Increment
433 U 1032 2
434 U 1033 2 IF (CMPLNO (LINNO, EDTSSL_LNO_VFCMAX) GTR 0)
435 U 1034 2 THEN
436 U 1035 2 BEGIN
437 U 1036 2 EDTSSFMT MSG (EDT$_SEQINCROV);
438 U 1037 2 RETURN (0)
439 U 1038 2 END;
440 U 1039 2
441 U 1040 2 DECR I FROM 9 TO 5 DO
442 U 1041 2 BEGIN
443 U 1042 2 SEQ_INCR = .SEQ_INCR*10;
444 U 1043 2 EDTSLDIV (LINNO, DIGIT, .1);
445 U 1044 2 SEQ_INCR = .SEQ_INCR + .DIGIT;
446 U 1045 2 END;
447 U 1046 2
448 U 1047 2 END;
449 U 1048 2
```



```
450 1049 2 1+ Try to open the file and set up file specific messages
451 1050 2 1-
452 1051 2 1-
453 1052 2 1-
454 1053 2 1-
455 1054 2 1-
456 1055 2 1-
457 1056 2 1-
458 1057 2 1-
459 1058 2 1-
460 1059 2 1-
461 1060 2 1-
462 1061 2 1-
463 1062 2 1-
464 1063 2 1-
465 1064 2 1-
466 1065 2 1-
467 1066 2 1-
468 1067 2 1-
469 1068 2 1-
470 1069 2 1-
471 1070 2 1-
472 1071 2 1-
473 1072 2 1-
474 1073 2 1-
475 1074 2 1-
476 1075 2 1-
477 1076 2 1-
478 1077 2 1-
479 1078 2 1-
480 1079 2 1-
481 1080 2 1-
482 1081 2 1-
483 1082 2 1-
484 1083 2 1-
485 1084 2 1-
486 1085 2 1-
487 1086 2 1-
488 1087 2 1-
489 1088 2 1-
490 1089 2 1-
491 1090 2 1-
492 1091 2 1-
493 1092 2 1-
494 1093 2 1-
495 1094 2 1-
496 1095 2 1-
497 1096 2 1-
498 1097 2 1-
499 1098 2 1-
500 1099 2 1-
501 1100 2 1-
502 1101 2 1-
503 1102 2 1-
504 1103 2 1-
505 1104 2 1-
506 1105 2 1-

1+ Try to open the file and set up file specific messages
1-
IF (.CHECK EQL 0)
THEN
BEGIN
FILESTRM = EDT$K_WRITE_FILE;
CLOSEMSG = EDT$QWIFILCLO;
OPNMSG = EDT$_WRIFILCRE;
END
ELSE
BEGIN
FILESTRM = EDT$K_OUTPUT_FILE;
CLOSEMSG = EDT$_OUTFILCLO;
OPNMSG = EDT$_OUTFILCRE;
END;

IF (.SEQ NEQ 0) THEN FILECODE = EDT$K_OPEN_OUTPUT_SEQ ELSE FILECODE = EDT$K_OPEN_OUTPUT_NOSEQ;
STRING_DESC (FILE_DESC, NAME_LEN, .NAME);
STATUS = EDT$$CALCFIO (.FILECODE, .FILESTRM, FILE_DESC, RHB_DESC);

IF .STATUS
THEN
BEGIN
MOVELINE (EDT$$L_LNO_ZERO, L_COUNT);
C_COUNT = 0;
EDT$$G_ABT_WRITE = 0;

WHILE (EDT$$NXT_LNRNG (0) AND (.EDT$$A_WK_LN NEQA EDT$$Z_EOB_LN) AND ( NOT .EDT$$G_ABT_WRITE)) DO
BEGIN
1+ Check for a CONTROL/C. If one has been typed abort the write operation.
1-
IF EDT$$CHK_CC ()
THEN
BEGIN
EDT$$G_CC_DONE = 1;
EDT$$G_ABT_WRITE = 1;
END
ELSE
BEGIN
1+ Set up the RHB if /SEQ was used.
1-
IF (.SEQ NEQ 0)
THEN
IF (.EDT$$Z_EXE_SBLK [SEQ_VAL] EQL 0) !
THEN
1+ If no sequence start was given, then use the buffer's current line
numbers in the VFC field (/SEQ)
1-
```

EDT\$WRITE
V04-000

EDT\$WRITE - write to a file
WRITE_FILE - write on a file

K 1
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1
Page 10
(3)

```

507 1106 6 BEGIN
508 1107 6 MOVE LINE (EDT$SA_WK_LN [LIN_NUM], LINNO);
509 1108 6
510 1109 6 DECR I FROM 14 TO 10 DO
511 1110 6
512 1111 6 WHILE (CMLNO (LINNO, EDT$SL_LNOO [.I]) GEQ 0) DO
513 1112 6 SUBLINE (EDT$SL_LNOO [.I], LINNO);
514 1113 6
515 1114 7 IF (CMLNO (LINNO, EDT$SL_LNO_VFCMAX) GEQ 0)
516 1115 6 THEN
517 1116 6 EDT$SL_IO_VFCHD = 65535
518 1117 6 ELSE
519 1118 7 BEGIN
520 1119 7 !+
521 1120 7 !- Set up sequence numbers, but first divide by 10**5
522 1121 7
523 1122 7 EDT$SL_IO_VFCHD = 0;
524 1123 7
525 1124 7 DECR I FROM 9 TO 5 DO
526 1125 8 BEGIN
527 1126 8 EDT$SL_IO_VFCHD = .EDT$SL_IO_VFCHD*10;
528 1127 8 EDT$SLDIV (LINNO, DIGIT, -1);
529 1128 8 EDT$SL_IO_VFCHD = .EDT$SL_IO_VFCHD + .DIGIT;
530 1129 8 END
531 1130 8
532 1131 7 END
533 1132 7
534 1133 6 ELSE
535 1134 5 !+
536 1135 5 !- Otherwise, use the given sequence start and increment (/SEQ:st:inc)
537 1136 5
538 1137 5 BEGIN
539 1138 6 EDT$SL_IO_VFCHD = .SEQ_NUM;
540 1139 6 SEQ_NUM = .SEQ_NUM + .SEQ_INCR;
541 1140 6
542 1141 6 IF (.SEQ_NUM EQL 65535) THEN SEQ_INCR = 0;
543 1142 6
544 1143 6 END;
545 1144 5
546 1145 5 !+
547 1146 5 !- Write a line to the file.
548 1147 5
549 1148 5
550 1149 5 RHB_DESC [DSC$W_LENGTH] = 2;
551 1150 5 RHB_DESC [DSC$A-POINTER] = EDT$SL_IO_VFCHD;
552 1151 5 STAT_DESC [DSC$W_LENGTH] = .EDT$SA_WK_LN [LIN_LENGTH];
553 1152 5 STAT_DESC [DSC$A-POINTER] = EDT$SA_WK_LN [LIN_TEXT];
554 1153 5 STATUS = EDT$SCALFIO (EDT$K_PUT, .FICESTRM, STAT_DESC, RHB_DESC);
555 1154 5
556 1155 6 IF ( NOT .STATUS)
557 1156 5 THEN
558 1157 6 BEGIN
559 1158 6
560 L 1159 6 !IF XBLISS (BLISS32)
561 1160 6 !THEN
562 1161 6 EDT$FIOPN_ERR (EDT$_ERROUTFIL, .EDT$SA_IO_FNAM);
563 1162 6 !FI
```


EDT\$WRITE
V04-000

EDT\$WRITE - write to a file
WRITE_FILE - write on a file

L 1
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1 Page 11
(3)

```

564 1163 6
565 1164 6 EDT$G_ABT_WRITE = 1;
566 1165 6 END
567 1166 6 ELSE
568 1167 6 BEGIN
569 1168 6 +
570 1169 6 - Count lines and characters for consistency check.
571 1170 6
572 1171 6 ADDLINE (NUMBER ONE, L COUNT);
573 1172 6 C COUNT = .C_COUNT + .EDT$A_WK_LN [LIN_LENGTH];
574 1173 6 END;
575 1174 6
576 1175 6 END;
577 1176 6
578 1177 6 END;
579 1178 6
580 1179 6 +
581 1180 6 - Watch for the write abort flag, which can be set if we have an error on the input file,
582 1181 6 by control C or by an error writing the output file.
583 1182 6
584 1183 6
585 1184 6 IF .EDT$G_ABT_WRITE
586 1185 6 THEN
587 1186 6 BEGIN
588 1187 6 STATUS = EDT$CALLFID (EDT$K_CLOSE_DEL, .FILESTRM, FILE_DESC, 0);
589 1188 6
590 1189 6 IF ( NOT .STATUS) THEN EDT$FIOPN_ERR (.CLOSEMSG, FILE_DESC);
591 1190 6
592 1191 6 EDT$FMT_MSG (EDT$NOFILWRT);
593 1192 6
594 1193 6 L XIF XBLISS (BLISS32)
595 1194 6 XTHEN
596 1195 6 STR$FREE1_DX (FILE_DESC);
597 1196 6 XFI
598 1197 6
599 1198 6 RETURN (0);
600 1199 6 END;
601 1200 6
602 1201 6 +
603 1202 6 - Do the consistency check.
604 1203 6
605 1204 6
606 1205 6 IF (.CHECK NEQ 0)
607 1206 6 THEN
608 1207 6
609 1208 6 IF ( NOT (LINNOEQL (L COUNT, EDT$A_CUR_BUF [TBCB_LINE_COUNT])) OR !
610 1209 6 (.C_COUNT NEQ .EDT$A_CUR_BUF [TBCB_CHAR_COUNT]))
611 1210 6 THEN
612 1211 6 BEGIN
613 1212 6 EDT$FMT_MSG (EDT$CONCHKFLD);
614 1213 6 EDT$G_EXE_SBITS = (.EDT$G_EXE_SBITS OR OPT_SAVE);
615 1214 6 END;
616 1215 6
617 1216 6 +
618 1217 6 - Close the output file and print a message giving number of lines written
619 1218 6 to the file.
620 1219 6
```

EDT\$WRITE
V04-000

EDT\$WRITE - write to a file
WRITE_FILE - write on a file

M 1
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1
Page 12
(3)

```

621 1220 3
622 1221 3      IF (.EDT$$G_EXT_MOD) THEN EDT$$STOP_WKINGMSG ();
623 1222 3
624 1223 3      STATUS = EDT$$CALLFIO (EDT$K_CLOSE, .FILESTRM, FILE_DESC, 0);
625 1224 3
626 1225 3      IF .STATUS
627 1226 3      THEN
628 1227 3          BEGIN
629 1228 4
630 1229 4      !+ Reset the screen width on EXIT if necessary. If the screen was reset, then
631 1230 4      !- reposition the cursor at the bottom of the screen.
632 1231 4
633 1232 4
634 1233 3          IF (.CHECK_NEQ 0)
635 1234 4          THEN
636 1235 4
637 1236 3              IF (.EDT$$G_TI_WID NEQ .EDT$$G_SAV_TIWID)
638 1237 4              THEN
639 1238 5                  BEGIN
640 1239 5                      EDT$$SC_SETWID (.EDT$$G_SAV_TIWID);
641 1240 5                      EDT$$SC_POSCSIF (.EDT$$G_MESSAGE_LINE, 0);
642 1241 4                      END;
643 1242 4
644 1243 4      !+
645 1244 4      !- Print a message if one is requested. This will be the 'input file does not have standard format'
646 1245 4      !- message deferred because we thought the user was never going to write the buffer.
647 1246 4
648 1247 4
649 1248 3          IF (.EDT$$G_WRITE_MSG NEQ 1)
650 1249 4          THEN
651 1250 5              BEGIN
652 1251 5                  EDT$$FMT_MSG (.EDT$$G_WRITE_MSG);
653 1252 5                  EDT$$G_WRITE_MSG = 1;
654 1253 4                  END;
655 1254 4
656 1255 4  L  %IF SUPPORT_WPS
657 1256 4      %THEN
658 1257 4
659 1258 4          IF .EDT$$G_SUMRY
660 1259 4          THEN
661 1260 4      %FI
662 1261 4
663 1262 3          BEGIN
664 1263 3      !+
665 1264 3      !- Extract the resultant filename of the opened file and format
666 1265 3      !- it to write out if the summary flag is set.
667 1266 3
668 1267 3
669 1268 3  L  %IF %BLISS (BLISS32)
670 1269 3      %THEN
671 1270 3          EDT$$FMT_STR (.FILE_DESC [DSC$A_POINTER], .FILE_DESC [DSC$W_LENGTH]);
672 1271 3  U  %ELSE
673 1272 3      EDT$$GET_FNAM ();
674 1273 3      %FI
675 1274 3
676 1275 3          EDT$$FMT_CH (' ');
677 1276 3          EDT$$FMT_STRCNT (L_COUNT, UPLIT (' Line'), 5);
```



```

678      1277 5      EDT$FMT_CRLF ( )
679      1278 4      END;
680      1279 4
681      1280 4      !+
682      1281 4      ! On VMS deallocate all dynamic descriptors used
683      1282 4      !-
684      1283 4
685      L 1284 4      %IF %BLISS (BLISS32)
686      1285 4      %THEN
687      1286 4          STR$FREE1_DX (FILE_DESC);
688      1287 4      %FI
689      1288 4
690      1289 4          RETURN (1);
691      1290 4      END
692      1291 3      ELSE
693      1292 4          BEGIN
694      1293 4      !+
695      1294 4      ! File was not closed, output error message.
696      1295 4      !-
697      1296 4          EDT$FIOPN_ERR (.CLOSEMSG, FILE_DESC);
698      1297 4      !+
699      1298 4      ! On VMS deallocate all dynamic descriptors used
700      1299 4      !-
701      1300 4
702      L 1301 4      %IF %BLISS (BLISS32)
703      1302 4      %THEN
704      1303 4          STR$FREE1_DX (FILE_DESC);
705      1304 4      %FI
706      1305 4
707      1306 4          RETURN (0);
708      1307 4      END
709      1308 4
710      1309 3      END
711      1310 2      ELSE
712      1311 4          BEGIN
713      1312 4      !+
714      1313 4      ! File was not created, output error message.
715      1314 4      !-
716      1315 4          EDT$FIOPN_ERR (.OPNMSG, FILE_DESC);
717      1316 4      !+
718      1317 4      ! On VMS deallocate all dynamic descriptors used
719      1318 4      !-
720      1319 4
721      L 1320 4      %IF %BLISS (BLISS32)
722      1321 4      %THEN
723      1322 4          STR$FREE1_DX (FILE_DESC);
724      1323 4      %FI
725      1324 4
726      1325 4          RETURN (0);
727      1326 4      END
728      1327 4
729      1328 1      END;
```

! of routine WRITE_FILE

.TITLE EDTSLWRITE EDTSLWRITE - write to a file
.IDENT \V04-000\

EDTSLWRITE
V04-000

EDTSLWRITE - write to a file
WRITE_FILE - write on a file

B 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1

Page 14
(3)

```
.PSECT _EDT$CODE,NOWRT, SHR, PIC,2
00 00 00 65 6E 69 6C 20 00000 P.AAA: .ASCII \ line\<0><0><0>
      :
      .EXTRN EDT$M_NOOUTPUT, EDT$K_WRITE_FILE
      .EXTRN EDT$K_OUTPUT_FILE
      .EXTRN EDT$K_OPEN_OUTPUT_SEQ
      .EXTRN EDT$K_OPEN_OUTPUT_NOSEQ
      .EXTRN EDT$K_PUT, EDT$K_CLOSE
      .EXTRN EDT$K_CLOSE_DEL
      .EXTRN EDT$$SC_SETQID, EDT$$SC_POSCSIF
      .EXTRN EDT$$STOP_WKINGMSG
      .EXTRN EDT$$CHK_CC, EDT$$LDIV
      .EXTRN EDT$$CALCFIO, EDT$$FMT_CH
      .EXTRN EDT$$FMT_STRCNT
      .EXTRN EDT$$FMT_CRLF, EDT$$FMT_MSG
      .EXTRN EDT$$NXT_LNRNG, EDT$$RNG_REPOS
      .EXTRN EDT$$FIOPN_ERR, EDT$$FMT_STR
      .EXTRN STR$FREE1_DX, EDT$$G_TI_QID
      .EXTRN EDT$$G_ABT_WRITE
      .EXTRN EDT$$G_SAV_TI_QID
      .EXTRN EDT$$G_MESSAGE_LINE
      .EXTRN EDT$$A_IO_FNAM, EDT$$G_EXT_MOD
      .EXTRN EDT$$G_OUT_NAMLEN
      .EXTRN EDT$$A_OUT_NAM, EDT$$G_RNG_FRSTLN
      .EXTRN EDT$$Z_RNG_ORIGPOS
      .EXTRN EDT$$V_OPTIONS, EDT$$A_CUR_BUF
      .EXTRN EDT$$L_IO_VFCHD
      .EXTRN EDT$$L_LNO_ZERO
      .EXTRN EDT$$L_LNO_VFCMAX
      .EXTRN EDT$$L_LNOD, EDT$$A_FMT_CUR
      .EXTRN EDT$$G_FMT_LNPOS
      .EXTRN EDT$$T_FMT_BUF, EDT$$G_SUMRY
      .EXTRN EDT$$Z_EOB_LN, EDT$$A_QK_LN
      .EXTRN EDT$$A_EXE_CURCMD
      .EXTRN EDT$$G_EXE_SBITS
      .EXTRN EDT$$Z_EXE_SBLK
      .EXTRN EDT$$G_CC_DONE, EDT$$G_WRITE_MSG
      .EXTRN EDT$_NOFILCSPC, EDT$_NOFILWRT
      .EXTRN EDT$_CONCHKFLD, EDT$_ERROUTFIL
      .EXTRN EDT$_WRIFILCRE, EDT$_OUTFILCRE
      .EXTRN EDT$_SEQNUMOV, EDT$_SEQINCROV
      .EXTRN EDT$_OUTFILCLO, EDT$_WRIFILCLO
      .EXTRN STR$COPY_R
```

```
OFFC 00000 WRITE_FILE:
      .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 0765
      .SUBL2 #56, SP
      .MOVL #17694720, RHB_DESC : 0929
      .MOVW #270, STAT_DESC+2 : 0922
      .MOVL #34471936, FILE_DESC : 0931
      .CLRL RHB_DESC+4 : 0928
      .CLRL FILE_DESC+4 : 0930
      .CLRL NAME_LEN : 0933
      .MOVL #1, EDT$$G_RNG_FRSTLN : 0937
      .MOVL EDT$$A_CUR_BUF, R0 : 0938
      .MOVC3 #14, (R0), EDT$$Z_RNG_ORIGPOS
      5E 010E0000 38 C2 00002
      28 AE 010E0000 8F D0 00005
      22 AE 010E 8F B0 0000D
      30 AE 020E0000 8F D0 00013
      2C AE D4 0001B
      34 AE D4 0001E
      08 AE D4 00021
      00000000G 00 01 D0 00024
      50 00000000G 00 D0 0002B
      00000000G 00 60 0E 28 00032
```


EDTSLWRITE
V04-000

EDTSLWRITE - write to a file
WRITE_FILE - write on a file

C 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1

Page 15
(3)

ED
VO

00000000G	00	04	AC	DD	0003A	PUSHL	RANGE	0940
	03		01	FB	0003D	CALLS	#1, EDT\$SRNG_REPOS	
			50	E8	00044	BLBS	R0, 1\$	
		0484	31	00047	BRW	68\$		
	50	00000000G	00	D0	0004A	1\$:	EDT\$SA_EXE_CURCMD, R0	0946
		0C	A0	D5	00051	TSTL	12(R0)	
			09	13	00054	BEQL	2\$	
	5A	08	A0	D0	00056	MOVL	8(R0), NAME	0949
08	AE	0C	A0	D0	0005A	MOVL	12(R0), NAME_LEN	0950
	56	08	AC	D0	0005F	2\$:	MOVL CHECK, R6	0957
		04	AE	D4	00063	CLRL	4(SP)	
			56	D5	00066	TSTL	R6	
			17	13	00068	BEQL	3\$	
		04	AE	D6	0006A	INCL	4(SP)	
		08	AE	D5	0006D	TSTL	NAME_LEN	
			0F	12	00070	BNEQ	3\$	
	5A	00000000G	00	D0	00072	MOVL	EDT\$SA_OUT_NAM, NAME	0960
08	AE	00000000G	00	D0	00079	MOVL	EDT\$SG_OUT_NAMLEN, NAME_LEN	0961
		08	AE	D5	00081	3\$:	TSTL NAME_LEN	0975
			19	12	00084	BNEQ	5\$	
			56	D5	00086	TSTL	R6	
			0D	13	00088	BEQL	4\$	
00000000G	8F	00000000G	00	D3	0008A	BITL	EDT\$SV_OPTIONS, #EDT\$M_NOOUTPUT	
			08	13	00095	BEQL	5\$	
		00000000G	8F	DD	00097	4\$:	PUSHL #EDT\$NOFILSPC	0978
			52	11	0009D	BRB	11\$	
57	00000000G	00	01	04	EF	5\$:	EXTZV #4, #1, EDT\$SG_EXE_SBITS, SEQ	1004
			03	57	E8	000A8	BLBS	SEQ, 6\$
				00CE	31	000AB	BRW	21\$
			59	D4	000AE	6\$:	CLRL SEQ_NUM	1010
			5B	D4	000B0	CLRL	SEQ_INCR	1011
			00	D0	000B2	MOVL	EDT\$SZ_EXE_SBLK, R0	1012
10	AE	08	50	00000000G	00	28	000B9	
			50	00000000G	00	3C	000BF	
			50	14	AE	B1	000C6	
				0F	1F	000CA	BLSSU	7\$
				18	12	000CC	BNEQ	9\$
			50	00000000G	00	D0	000CE	
			50	10	AE	D1	000D5	
				05	1E	000D9	BGEQU	8\$
			50	01	CE	000DB	7\$:	MNEGL #1, R0
				09	11	000DE	BRB	10\$
				04	12	000E0	8\$:	BNEQ 9\$
			50	D4	000E2	CLRL	R0	
				03	11	000E4	BRB	10\$
			50	01	D0	000E6	9\$:	MOVL #1, R0
				08	15	000E9	10\$:	BLEQ 12\$
				8F	DD	000EB	PUSHL	#EDT\$SEQNUMOV
		00000000G	5F	11	000F1	11\$:	BRB	18\$
			52	09	D0	000F3	12\$:	MOVL #9, I
			59	0A	C4	000F6	13\$:	MULL2 #10, SEQ_NUM
				52	DD	000F9	PUSHL	I
				10	AE	9F	000FB	PUSHAB
				18	AE	9F	000FE	PUSHAB
				03	FB	00101	CALLS	#3, EDT\$SLDIV
		00000000G	00	59	0C	AE	C0	00108
FFE3	52	FF	8F	05	9D	0010C	ACBB	#5, #-1, I, 13\$

EDTSLWRITE
V04-000

EDTSLWRITE - write to a file
WRITE_FILE - write on a file

D 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1

Page 16
(3)

ED
V0

10	AE	14	50	00000000G	00	D0	00113	MOVL	EDT\$\$Z_EXE_SBLK, R0	1031	
			A0		06	28	0011A	MOVCL	#6, 20(R0); LINNO		
			50	00000000G	00	3C	00120	MOVZWL	HIGH_2, R0	1033	
			50	14	AE	B1	00127	CMPL	HIGH_1, R0		
					0F	1F	0012B	BLSSU	14\$		
					18	12	0012D	BNEQ	16\$		
			50	00000000G	00	D0	0012F	MOVL	LOW_2, R0		
			50	10	AE	D1	00136	CMPL	LOW_1, R0		
					05	1E	0013A	BGFQU	15\$		
			50		01	CE	0013C	MNEGL	#1, R0		
					09	11	0013F	BRB	17\$		
					04	12	00141	BNEQ	16\$		
					50	D4	00143	CLRL	R0		
					03	11	00145	BRB	17\$		
			50		01	D0	00147	MOVL	#1, R0		
					10	15	0014A	BLEQ	19\$		
			00000000G	00	8F	DD	0014C	PUSHL	#EDT\$, SEQINCR	1036	
					01	FB	00152	CALLS	#1, EDT\$\$FMT_MSG		
					0372	31	00159	BRW	68\$	1037	
			52		09	D0	0015C	MOVL	#9, I	1040	
			5B		0A	C4	0015F	MULL2	#10, SEQ_INCR	1042	
					52	DD	00162	PUSHL	I	1043	
				10	AE	9F	00164	PUSHAB	DIGIT		
				18	AE	9F	00167	PUSHAB	LINNO		
			00000000G	00	03	FB	0016A	CALLS	#3, EDT\$\$LDIV		
			5B	0C	AE	C0	00171	ADDL2	DIGIT, SEQ_INCR	1044	
FFE3			52	FF	8F	05	9D	00175	ACBB	#5, #-1, I, 20\$	1040
					56	D5	0017C	TSTL	R6	1053	
					17	12	0017E	BNEQ	22\$		
			56	00000000G	8F	D0	00180	MOVL	#EDT\$K_WRITE_FILE, FILESTRM	1056	
			58	00000000G	8F	D0	00187	MOVL	#EDT\$K_ORIFILCLO, CLOSEMSG	1057	
			53	00000000G	8F	D0	0018E	MOVL	#EDT\$K_WRIFILCRE, OPNMSG	1058	
					15	11	00195	BRB	23\$	1053	
			56	00000000G	8F	D0	00197	MOVL	#EDT\$K_OUTPUT_FILE, FILESTRM	1062	
			58	00000000G	8F	D0	0019E	MOVL	#EDT\$K_OUTFILCLO, CLOSEMSG	1063	
			53	00000000G	8F	D0	001A5	MOVL	#EDT\$K_OUTFILCRE, OPNMSG	1064	
					6E	D4	001AC	CLRL	(SP)	1067	
					57	D5	001AE	TSTL	SEQ		
					0B	13	001B0	BEQL	24\$		
					6E	D6	001B2	INCL	(SP)		
			52	00000000G	8F	D0	001B4	MOVL	#EDT\$K_OPEN_OUTPUT_SEQ, FILECODE		
					07	11	001BB	BRB	25\$		
			52	00000000G	8F	D0	001BD	MOVL	#EDT\$K_OPEN_OUTPUT_NOSEQ, FILECODE		
					5A	DD	001C4	PUSHL	NAME	1069	
				0C	AE	9F	001C6	PUSHAB	NAME_LEN		
				38	AE	9F	001C9	PUSHAB	FILE_DESC		
			00000000G	00	03	FB	001CC	CALLS	#3, STR\$COPY_R		
					28	AE	9F	001D3	PUSHAB	RHB_DESC	1070
					34	AE	9F	001D6	PUSHAB	FILE_DESC	
					0044	8F	BB	001D9	PUSHR	#4M<R2,R6>	
			00000000G	00	04	FB	001DD	CALLS	#4, EDT\$\$CALLF10		
				57	50	D0	001E4	MOVL	R0, STATUS		
				03	57	E8	001E7	BLBS	STATUS, 26\$	1072	
					02CB	31	001EA	BRW	65\$		
18	AE	00000000G	00		06	28	001ED	MOVCL	#6, EDT\$\$L_LNO_ZERO, L_COUNT	1075	
					5A	D4	001F6	CLRL	C_COUNT	1076	
			00000000G	00	D4	001F8	CLRL	EDT\$\$G_ABT_WRITE		1077	

				7E	D4	001FE	27\$:	CLRL	-(SP)		1079		
				01	F8	00200		CALLS	#1, EDT\$\$NXT_LNRNG				
				50	E8	00207		BLBS	R0, 29\$				
				017D	31	0020A	28\$:	BRW	53\$				
				50	9E	0020D	29\$:	MOVAB	EDT\$\$Z_EOB_LN, R0				
				50	00000000G	00		CMPL	EDT\$\$A_WK_CN, R0				
					ED	13		BEQL	28\$				
				03	00000000G	00		BLBC	EDT\$\$G_ABT_WRITE, 30\$				
					016A	31		BRW	54\$				
				00	00	F8	00227	30\$:	CALLS	#0, EDT\$\$CHK_CC	1085		
				0A	50	E9	0022E		BLBC	R0, 31\$			
				00	01	D0	00231		MOVL	#1, EDT\$\$G_CC_DONE	1088		
					012E	31	0023B		BRW	49\$	1089		
				03	6E	E8	0023B	31\$:	BLBS	(SP), 32\$	1097		
					00DE	31	0023E		BRW	48\$			
				50	00000000G	00	D0	00241	32\$:	MOVL	EDT\$\$Z_EXE_SBLK, R0	1100	
					01	A0	95	00248		TSTB	1(R0)		
					03	13	0024B		BEQL	33\$			
					00BA	31	0024D		BRW	47\$			
				50	00000000G	00	D0	00250	33\$:	MOVL	EDT\$\$A_WK_LN, R0	1107	
				A0	06	28	00257		MOV C3	#6, 1(R0), LINNO			
				50	0E	D0	0025D		MOVL	#14, I	1109		
				50	06	C5	00260	34\$:	MULL3	#6, I, R1	1111		
				51	00000000G	0041	9E	00264		MOVAB	EDT\$\$L_LN00[R1], R1		
				04	A1	14	AE	B1	0026C	35\$:	CMPW	HIGH_1, 4(R1)	
						08	1F	00271		BLSSU	36\$		
						11	12	00273		BNEQ	38\$		
				61	10	AE	D1	00275		CMPL	LOW_1, (R1)		
						05	1E	00279		BGEQU	37\$		
				52		01	CE	0027B	36\$:	MNEGL	#1, R2		
						09	11	0027E		BRB	39\$		
						04	12	00280	37\$:	BNEQ	38\$		
						52	D4	00282		CLRL	R2		
						03	11	00284		BRB	39\$		
				52		01	D0	00286	38\$:	MOVL	#1, R2		
						13	19	00289	39\$:	BLSS	40\$		
				52	16	AE	B0	0028B		MOVW	UPPER_WORD, SAVE	1112	
				10		61	C2	0028F		SUBL2	(R1)-LINNO		
				14		A1	D9	00293		SBWC	4(R1), LINNO		
				16		52	B0	00298		MOVW	SAVE, UPPER_WORD		
						CE	11	0029C		BRB	35\$	1111	
				FF		0A	9D	0029E	40\$:	ACBB	#10, #-1, I, 34\$	1114	
				8F	00000000G	00	3C	002A5		MOVZWL	HIGH_2, R0		
				50		AE	B1	002AC		CMPW	HIGH_1, R0		
				50	14	0F	1F	002B0		BLSSU	41\$		
						18	12	002B2		BNEQ	43\$		
				50	00000000G	00	D0	002B4		MOVL	LOW_2, R0		
				50	10	AE	D1	002BB		CMPL	LOW_1, R0		
						05	1E	002BF		BGEQU	42\$		
				50		01	CE	002C1	41\$:	MNEGL	#1, R0		
						09	11	002C4		BRB	44\$		
						04	12	002C6	42\$:	BNEQ	43\$		
						50	D4	002C8		CLRL	R0		
						03	11	002CA		BRB	44\$		
				50		01	D0	002CC	43\$:	MOVL	#1, R0		
						09	19	002CF	44\$:	BLSS	45\$		
				00000000G	00	01	AE	002D1		MNEGW	#1, EDT\$\$L_IO_VFCHD	1116	

Address	Hex	Assembly	Comment
00000000G	00	BRB	48\$
00000000G	00	EDTSSL_ID_VFCHD	1122
00000000G	00	CLRW	1124
00000000G	00	MOVW	1126
00000000G	00	MULW2	1127
00000000G	00	PUSHL	
00000000G	00	PUSHAB	DIGIT
00000000G	00	PUSHAB	LINNO
00000000G	00	CALLS	#3, EDTSSLDIV
00000000G	00	ADDW2	DIGIT, EDTSSL_ID_VFCHD
00000000G	00	ACBB	#5, #-1, 1, 48\$
00000000G	00	BRB	48\$
00000000G	00	MOVW	SEQ_NUM, EDTSSL_ID_VFCHD
00000000G	00	ADDL2	SEQ_INCR, SEQ_NUM
00000000G	00	CMPL	SEQ_NUM, #65535
00000000G	00	BNEQ	48\$
00000000G	00	CLRL	SEQ_INCR
00000000G	00	MOVW	#2, RHB_DESC
00000000G	00	MOVAB	EDTSSL_ID_VFCHD, RHB_DESC+4
00000000G	00	MOVL	EDTSSA_WK_LN, RO
00000000G	00	MOVZBW	(RO), STAT_DESC
00000000G	00	MOVAB	7(RO), STAT_DESC+4
00000000G	00	PUSHAB	RHB_DESC
00000000G	00	PUSHAB	STAT_DESC
00000000G	00	PUSHL	FILESTRM
00000000G	00	PUSHL	#EDTSK_PUT
00000000G	00	CALLS	#4, EDTSSCALLFIO
00000000G	00	MOVL	RO, STATUS
00000000G	00	BLBS	STATUS, 50\$
00000000G	00	PUSHL	EDTSSA_ID_FNAM
00000000G	00	PUSHL	#EDTS_ERROUTFIL
00000000G	00	CALLS	#2, EDTSSFIOPN_ERR
00000000G	00	MOVL	#1, EDTSSG_ABT_WRITE
00000000G	00	BRB	52\$
00000000G	00	INCL	FIRST_LWORD
00000000G	00	BNEQ	51\$
00000000G	00	INCL	NEXT_WORD
00000000G	00	MOVL	EDTSSA_WK_LN, RO
00000000G	00	MOVZBL	(RO), R1
00000000G	00	ADDL2	R1, C_COUNT
00000000G	00	BRW	27\$
00000000G	00	BLBC	EDTSSG_ABT_WRITE, 56\$
00000000G	00	CLRL	-(SP)
00000000G	00	PUSHAB	FILE_DESC
00000000G	00	PUSHL	FILESTRM
00000000G	00	PUSHL	#EDTSK_CLOSE_DEL
00000000G	00	CALLS	#4, EDTSSCALCFIO
00000000G	00	MOVL	RO, STATUS
00000000G	00	BLBS	STATUS, 55\$
00000000G	00	PUSHAB	FILE_DESC
00000000G	00	PUSHL	CLOSEMSG
00000000G	00	CALLS	#2, EDTSSFIOPN_ERR
00000000G	00	PUSHL	#EDTS_NOFILWRT
00000000G	00	CALLS	#1, EDTSSFMT_MSG
00000000G	00	BRW	67\$
00000000G	00	BLBC	4(SP), 58\$
00000000G	00	MOVL	EDTSSA_CUR_BUF, RO
00000000G	00	CMPL	LOW_1, -24(RO)

EDTSLWRITE
V04-000

EDTSLWRITE - write to a file
WRITE_FILE - write on a file

G 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1

Page 19
(3)

1C	A0	1C	0D	12	003D7	BNEQ	57\$		
			AE	B1	003D9	CMPW	HIGH_1, 28(R0)		
1E	A0		06	12	003DE	BNEQ	57\$		
			5A	D1	003E0	CMPL	C COUNT, 30(R0)	1209	
		00000000G	15	13	003E4	BEQL	58\$		
00000000G	00		8F	DD	003E6	PUSHL	#EDT\$ CONCHKFLD	1212	
00000000G	00	40	01	FB	003EC	CALLS	#1, EDT\$\$FMT MSG		
	07	00000000G	8F	88	003F3	BISB2	#64, EDT\$\$G EXE_SBITS	1213	
00000000G	00		00	E9	003FB	BLBC	EDT\$\$G EXT MOD, -59\$	1221	
			00	FB	00402	CALLS	#0, EDT\$\$STOP_WKINGMSG		
		34	7E	D4	00409	CLRL	-(SP)	1223	
			AE	9F	0040B	PUSHAB	FILE_DESC		
		00000000G	56	DD	0040E	PUSHL	FILESTRM		
00000000G	00		8F	DD	00410	PUSHL	#EDT\$K CLOSE		
	57		04	FB	00416	CALLS	#4, EDT\$\$CALLF10		
	03		50	D0	0041D	MOVL	R0, STATUS		
			57	E8	00420	BLBS	STATUS, 60\$	1225	
	28	04	008B	31	00423	BRW	64\$		
	50	00000000G	AE	E9	00426	BLBC	4(SP), 61\$	1233	
	50	00000000G	00	D0	0042A	MOVL	EDT\$\$G_SAV TIWID, R0	1236	
			00	D1	00431	CMPL	EDT\$\$G_TI_WID, R0		
			18	13	00438	BEQL	61\$		
00000000G	00		50	DD	0043A	PUSHL	R0	1239	
			01	FB	0043C	CALLS	#1, EDT\$\$SC_SETWID		
		00000000G	7E	D4	00443	CLRL	-(SP)	1240	
00000000G	00		00	DD	00445	PUSHL	EDT\$\$G MESSAGE LINE		
	50	00000000G	02	FB	0044B	CALLS	#2, EDT\$\$SC_POSCSIF		
	01		00	D0	00452	MOVL	EDT\$\$G_WRITE_MSG, R0	1248	
			50	D1	00459	CMPL	R0, #1		
			10	13	0045C	BEQL	62\$		
00000000G	00		50	DD	0045E	PUSHL	R0	1251	
00000000G	00		01	FB	00460	CALLS	#1, EDT\$\$FMT MSG		
	2E	00000000G	01	D0	00467	MOVL	#1, EDT\$\$G_WRITE_MSG	1252	
	7E		00	E9	0046E	BLBC	EDT\$\$G_SUMRY, 63\$	1258	
		30	AE	3C	00475	MOVZWL	FILE_DESC, -(SP)	1270	
		38	AE	DD	00479	PUSHL	FILE_DESC+4		
00000000G	00		02	FB	0047C	CALLS	#2, EDT\$\$FMT_STR		
			20	DD	00483	PUSHL	#3, EDT\$\$FMT_STRCNT	1275	
00000000G	00		01	FB	00485	CALLS	#1, EDT\$\$FMT_CH		
			05	DD	0048C	PUSHL	#5	1276	
		FB66	CF	9F	0048E	PUSHAB	P.AAA		
		20	AE	9F	00492	PUSHAB	L COUNT		
00000000G	00		03	FB	00495	CALLS	#3, EDT\$\$FMT_STRCNT		
00000000G	00		00	FB	0049C	CALLS	#0, EDT\$\$FMT_CRLF	1277	
		30	AE	9F	004A3	PUSHAB	FILE_DESC	1286	
00000000G	00		01	FB	004A6	CALLS	#1, STR\$FREE1_DX		
	50		01	D0	004AD	MOVL	#1, R0	1292	
			04	004B0	RET				
		30	AE	9F	004B1	PUSHAB	FILE_DESC	1296	
			58	DD	004B4	PUSHL	CLOSEMSG		
			05	11	004B6	BRB	66\$		
		30	AE	9F	004B8	PUSHAB	FILE_DESC	1315	
			53	DD	004BB	PUSHL	OPNMSG		
00000000G	00		02	FB	004BD	CALLS	#2, EDT\$\$F1OPN_ERR		
		30	AE	9F	004C4	PUSHAB	FILE_DESC	1322	
00000000G	00		01	FB	004C7	CALLS	#1, STR\$FREE1_DX		
			50	D4	004CE	CLRL	R0	1328	

ED
VO

EDTSLWRITE
V04-000

EDTSLWRITE - write to a file
WRITE_FILE - write on a file

H 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1
Page 20 (3)

04 00400

RET

; Routine Size: 1233 bytes, Routine Base: _EDT\$CODE + 0008

ED
VO

EDTSLWRITE
V04-000

EDTSLWRITE - write to a file
EDT\$SEXI_CMD - EXIT line-mode command

1 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1
Page 21
(4)

```

731 1329 1 %SBTTL 'EDT$SEXI_CMD - EXIT line-mode command'
732 1330 1
733 1331 1 GLOBAL ROUTINE EDT$SEXI_CMD ! EXIT line-mode command
734 1332 1 : NOVALUE =
735 1333 1
736 1334 1 ++
737 1335 1 FUNCTIONAL DESCRIPTION:
738 1336 1
739 1337 1 Command processing routine for exit. Switch to the main buffer,
740 1338 1 write the output file and set the EDT$SG_EXITD flag
741 1339 1 if it succeeded.
742 1340 1
743 1341 1 FORMAL PARAMETERS:
744 1342 1
745 1343 1 NONE
746 1344 1
747 1345 1 IMPLICIT INPUTS:
748 1346 1
749 1347 1 EDT$SG_RCOV_MOD
750 1348 1
751 1349 1 IMPLICIT OUTPUTS:
752 1350 1
753 1351 1 EDT$SG_EXITD
754 1352 1
755 1353 1 ROUTINE VALUE:
756 1354 1
757 1355 1 NONE
758 1356 1
759 1357 1 SIDE EFFECTS:
760 1358 1
761 1359 1 NONE
762 1360 1
763 1361 1 --
764 1362 1
765 1363 2 BEGIN
766 1364 2
767 1365 2 EXTERNAL ROUTINE
768 1366 2 EDT$FND_BUF;
769 1367 2
770 1368 2 EXTERNAL
771 1369 2 EDT$SG_EXITD,
772 1370 2 EDT$SG_RCOV_MOD;
773 1371 2
774 1372 2 LOCAL
775 1373 2 RANGE : NODE_BLOCK;
776 1374 2
777 1375 2 ++
778 1376 2 Ignore the EXIT command encountered during recovery.
779 1377 2 --
780 1378 2
781 1379 2 IF .EDT$SG_RCOV_MOD THEN RETURN;
782 1380 2
783 1381 2 ++
784 1382 2 Setup the range as WHOLE.
785 1383 2 --
786 1384 2 RANGE [NODE_TYPE] = RANGE NODE;
787 1385 2 RANGE [RAN_TYPE] = RAN_WHOLE;
```

ED
VO

EDTSLWRITE
V04-000

EDTSLWRITE - write to a file
EDT\$SEXI_CMD - EXIT line-mode command

J 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1

Page 22
(4)

```

: 788      1386 2      RANGE [PREV_RANGE] = 0;
: 789      1387 2      +
: 790      1388 2      - Position into the MAIN buffer.
: 791      1389 2      -
: 792      1390 2      EDT$$FND_BUF (UPLIT (%STRING ('MAIN')), 4);
: 793      1391 2      +
: 794      1392 2      - Attempt to write the file, and set the EXITED flag if it succeeds.
: 795      1393 2      -
: 796      1394 2      EDT$$G_EXITD = WRITE_FILE (RANGE, 1);
: 797      1395 1      END;
                                ! of routine EDT$SEXI_CMD
```

```

                                004D9
                                4E 49 41 4D 004DC P.AAB: .BLKB 3
                                                                .ASCII \MAIN\
                                                                ;
                                                                .EXTRN EDT$$FND_BUF, EDT$$G_EXITD
                                                                .EXTRN EDT$$G_RCOV_MOD
                                                                .ENTRY EDT$SEXI_CMD, Save nothing
                                                                : 1331
                                                                SUBL2 #32, SP
                                                                : 1379
                                                                BLBS EDT$$G_RCOV_MOD, 1$
                                                                : 1384
                                                                MOVW #2818, -RANGE
                                                                : 1386
                                                                CLRL RANGE+20
                                                                : 1390
                                                                PUSHL #4
                                                                PUSHAB P.AAB
                                                                CALLS #2, EDT$$FND_BUF
                                                                : 1394
                                                                PUSHL #1
                                                                PUSHAB RANGE
                                                                CALLS #2, WRITE_FILE
                                                                MOVL R0, EDT$$G_EXITD
                                                                : 1395
                                                                RET
```

; Routine Size: 50 bytes, Routine Base: _EDT\$CODE + 04E0

; 798 1396 1

EDT\$WRITE
V04-000

EDT\$WRITE - write to a file
EDT\$\$WR_CMD - WRITE line-mode command

K 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1
Page 23
(5)

```

800 1397 1 XSBTTL 'EDT$$WR_CMD - WRITE line-mode command'
801 1398 1
802 1399 1 GLOBAL ROUTINE EDT$$WR_CMD ! WRITE line-mode command
803 1400 1 : NOVALUE =
804 1401 1
805 1402 1 ++
806 1403 1 FUNCTIONAL DESCRIPTION:
807 1404 1
808 1405 1 Command processing routine for the write command. Look at the current range;
809 1406 1 if it is null, use the whole buffer. Then write the file.
810 1407 1
811 1408 1 FORMAL PARAMETERS:
812 1409 1
813 1410 1 NONE
814 1411 1
815 1412 1 IMPLICIT INPUTS:
816 1413 1
817 1414 1 EDT$$Z_RNG_ORIGPOS
818 1415 1 EDT$$A_EXE_CURCMD
819 1416 1
820 1417 1 IMPLICIT OUTPUTS:
821 1418 1
822 1419 1 EDT$$A_CUR_BUF
823 1420 1
824 1421 1 ROUTINE VALUE:
825 1422 1
826 1423 1 NONE
827 1424 1
828 1425 1 SIDE EFFECTS:
829 1426 1
830 1427 1 NONE
831 1428 1
832 1429 1 --
833 1430 1
834 1431 2 BEGIN
835 1432 2
836 1433 2 EXTERNAL ROUTINE
837 1434 2 EDT$$RD_CURLN;
838 1435 2
839 1436 2 EXTERNAL
840 1437 2 EDT$$A_CUR_BUF : REF TBCB_BLOCK,
841 1438 2 EDT$$Z_RNG_ORIGPOS : POS_BLOCK,
842 1439 2 EDT$$A_EXE_CURCMD : REF NODE_BLOCK; ! Pointer to the current command.
843 1440 2
844 1441 2 LOCAL
845 1442 2 SAV_BUF, ! address of original buffer
846 1443 2 RANGE : REF NODE_BLOCK;
847 1444 2
848 1445 2 SAV_BUF = .EDT$$A_CUR_BUF; ! save original address
849 1446 2 RANGE = .EDT$$A_EXE_CURCMD [RANGE1];
850 1447 2
851 1448 2 IF (.RANGE [RAN_TYPE] EQL RAN_NULL) THEN RANGE [RAN_TYPE] = RAN_WHOLE;
852 1449 2
853 1450 2 WRITE_FILE (.RANGE, 0);
854 1451 2 ++
855 1452 2 ! Reposition to the original line.
856 1453 2 --
```

ED
V0

EDT\$WRITE
V04-000

EDT\$WRITE - write to a file
EDT\$\$WR_CMD - WRITE line-mode command

L 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1
Page 24
(5)

```
: 857      1454 2      EDT$$A_CUR_BUF = .SAV_BUF;      ! first get the buffer address
: 858      1455 2      EDT$$CPY_MEM (POS_SIZE, EDT$$Z_RNG_ORIGPOS, EDT$$A_CUR_BUF);
: 859      1456 2      EDT$$RD_CURLN ();      ! do the positioning
: 860      1457 1      END;      ! of routine EDT$$WR_CMD
```

```
                                .EXTRN EDT$$RD_CURLN
                                .ENTRY EDT$$WR_CMD, Save R2,R3,R4,R5,R6
                                MOVAB EDT$$A_CUR_BUF, R6      : 1399
                                MOVL EDT$$A_CUR_BUF, SAV_BUF    : 1445
                                MOVL EDT$$A_EXE_CURCMD, R0      : 1446
                                MOVL 4(R0), RANGE
                                TSTB 1(RANGE)
                                BNEQ 1$
                                MOVB #11, 1(RANGE)
                                CLRL -(SP)
                                PUSHL RANGE
                                CALLS #2, WRITE_FILE
                                MOVL SAV_BUF, EDT$$A_CUR_BUF    : 1454
                                MOVL EDT$$A_CUR_BUF, R0         : 1455
                                MOVC3 #14, EDT$$Z_RNG_ORIGPOS, (R0)
                                CALLS #0, EDT$$RD_CURLN
                                RET                                : 1456
                                : 1457

                                007C 00000
                                56 00000000G 00 9E 00002
                                52 66 D0 00009
                                50 00000000G 00 D0 0000C
                                50 04 A0 D0 00013
                                01 A0 95 00017
                                04 12 0001A
                                01 A0 0B 90 0001C
                                7E D4 00020 1$:
                                50 DD 00022
                                FACD CF 02 FB 00024
                                66 52 D0 00029
                                50 66 D0 0002C
                                60 00000000G 00 0E 28 0002F
                                00000000G 00 00 FB 00037
                                04 0003E
```

; Routine Size: 63 bytes, Routine Base: _EDT\$CODE + 0512

```
: 861      1458 1
: 862      1459 1 !<BLF/PAGE>
```

EDT\$LWRITE
V04-000

EDT\$LWRITE - write to a file
EDT\$\$WR_CMD - WRITE line-mode command

M 2
16-Sep-1984 01:03:36
14-Sep-1984 12:23:47

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]LWRITE.BLI;1

Page 25
(6)

: 864 1460 1 END
: 865 1461 1
: 866 1462 0 ELUDOM

! of module EDT\$LWRITE

PSECT SUMMARY

: Name Bytes Attributes
: _EDT\$CODE 1361 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[EDT.SRC]EDT.L32;1	377	109	28	40	00:00.2
_\$255\$DUA28:[EDT.SRC]PSECTS.L32;1	2	1	50	7	00:00.1
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	7	0	581	00:04.1
_\$255\$DUA28:[EDT.SRC]SUPPORTS.L32;1	2	1	50	5	00:00.2

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACEBACK/LIS=LISS:LWRITE/OBJ=OBJ\$:LWRITE MSRC\$:LWRITE.BLI/UPDATE=(ENH\$:LWRITE)

: Size: 1346 code + 15 data bytes
: Run Time: 01:00.3
: Elapsed Time: 01:15.8
: Lines/CPU Min: 1454
: Lexemes/CPU-Min: 7386
: Memory Used: 354 pages
: Compilation Complete

0136

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0137 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LXCOMM00
LIS

MCGETLIN
LIS

MCRIGHT
LIS

MCCHANGE
LIS

MACCAL
LIS

NOOPEN
LIS

PRAPPNLM
LIS

MSGTXT
LIS

PRGETTOK
LIS

PRISTOK
LIS

LXPRINT
LIS

MCTOP
LIS

MCDOWN
LIS

MCLEFT
LIS

MCBOTTOM
LIS

PAUDIT
LIS

PRGETCHR
LIS

LXCOM
LIS

MAIN
LIS

MCUP
LIS